# A Study on Web Caching Architectures and Performance

Athena I. Vakali
Department of Informatics, Aristotle University
Thessaloniki, 54006, Greece
*email: avakali@csd.auth.gr*

and

George E. Pallis
Department of Informatics, Aristotle University
Thessaloniki, 54006, Greece
*email: gpallis@csd.auth.gr*

## Abstract

As World Wide Web usage has grown dramatically in recent years, so has grown the recognition that Web caches (especially proxy caches) will have an important role in reducing server loads, client request latencies, and network traffic. In this survey we present the most common architectures for web caching and their most important characteristics are outlined. These architectures include proxy caching, cooperative caching, adaptive caching, push caching and active caching. Furthermore, emphasis is given on the basic metrics and factors for evaluating proxy cache performance.

**Key Words :** Web Caching, proxy caching architecture, Web caching performance.

## 1 Introduction

The surge in popularity of the World Wide Web (WWW) has introduced new issues such as Internet traffic and bandwidth consumption. Recently, much research has focused on improving Web performance by reducing the bandwidth consumption and WWW traffic. It means that fewer requests and responses need to go over the network and fewer request for a server to handle. Despite the fact that there have been great efforts for this purpose the results are not sufficient. The most approaches have presented the Web caching as the most beneficial solution for Web performance improvement. Web caching systems can lead to significant bandwidth savings, higher content availability, reducing client latency and increasing servers scalability and availability.
Proxy caching has become a well established technique for enabling effective file delivery within the WWW architecture [1]. One drawback of caching is the potential of using an out-of-date object stored in a cache instead of fetching the current object from the origin server. The Web documents are cached either directly by the browser or by a proxy server which is located "close " to clients. In general, the cache is a software that is in charge of storing on disks, data elements that are accessed by a number of clients. A cache server has a fixed amount of storage and when this storage space fills , the cache must choose a set of objects to evict to make room for newly requested objects.
The cache replacement policy determines which objects should be removed from the cache. Cache replacement algorithms play a central role in the design of any caching component. These algorithms usually maximize the cache hit ratio (the number of times that objects in the cache are referenced) by attempting to cache the data items which are most likely to be referenced in the near future. Unfortunately, it is very difficult, if not impossible, to predict the future user needs. Furthermore, several approaches have been suggested for cache replacement [2, 3, 4, 5].
The structure of the survey is as follows. In section 2 an overview of Web Caching Schemes is presented, with emphasis on architectures which deal with WWW caching. In section 3 the parameters for evaluating proxy cache performance are discussed. Section 4 summarizes conclusions.

## 2 Web Caching Schemes- Caching Architectures

Web caching is the approach of temporary storage of web objects, such as HTML files, for later retrieval. Few approaches have been suggested for effective web caching schemes. Several web caching architectures appeared in [2, 6, 7, 8, 9, 10]. The next sections present the most important web caching architectures implemented in earlier research efforts.

### 2.1 Proxy caching

A proxy cache server receives HTTP requests from clients for a web object and if it finds the requested object in its cache, it returns the object to the user without disturbing the upstream network connection or destination server. If it is not available in the cache, the proxy attempts to fetch the object directly from the object's home server. Finally the originating server, which has the object, gets it, possibly deposits it and returns the object to the user. The benefits of proxy caching are supposed to reduce network traffic and reduce average latency. Proxy caches are often

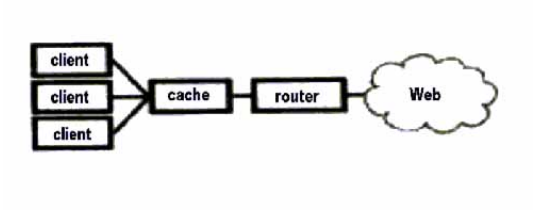


Figure 1: A standalone proxy configuration.

located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. When shared with other users, the proxies serve many clients with cached objects from many servers.
A standalone proxy configuration is shown in Figure 1. One disadvantage to this design is that the cache represents a single point of failure in the network. When the cache is unavailable , the network also appears unavailable to users. Furthermore, another drawback is that all user web browsers are manually configured to use the appropriate proxy cache. So, if the server is unavailable all of the users must reconfigure their browsers in order to use a different cache. A final issue related to the standalone approach is that there is no way to dynamically add more caches when needed. The Squid and the Microsoft Proxy Server are two proxies which are available as stand-alone systems.

### 2.1.1 Reverse Proxy Caching

An interesting variation to the proxy cache approach is the notion of reverse proxy caching, in which caches deployed near the servers, instead of near the clients. This is an attractive solution for servers that expect a high number of requests and want to assure a high level of quality of service (QoS). Reverse proxy caching is a useful mechanism when supporting virtual domains mapped to a single physical site, which is an popular service for many different service providers.

### 2.1.2 Transparent Caching

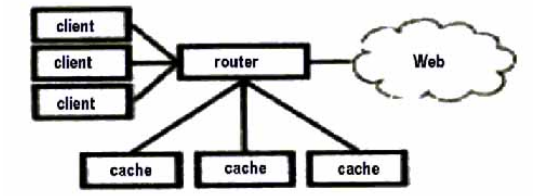


Figure 2: A router-transparent configuration.

One of the main drawbacks of the proxy server approach is the requirement to configure web browsers. The architecture of transparent caching eliminates this handicap. Transparent caches work by intercepting HTTP requests and redirecting them to web cache servers or clusters. There are two ways to deploy transparent proxy caching: at the switch level and at the router level. Router-based transparent proxy caching uses policy-based routing to direct requests to the appropriate cache or caches. For example, requests from certain clients can be associated with a particular cache. A router-transparent configuration is shown in Figure 2. In switch-based trans-

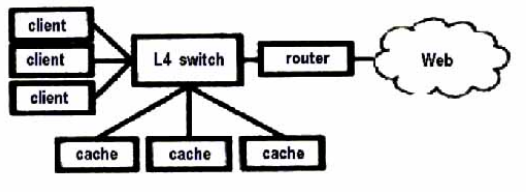


Figure 3: A switch transparent proxy caching configuration.

parent proxy caching the switch acts as a dedicated load balancer. This approach is attractive because it reduces the overhead normally incurred by policy-based routing. Although it adds extra cost to the deployment, switches are generally less expensive than routers. A switch transparent proxy caching configuration is shown in Figure 3. Note that L4 (Layer 4) switches rely on the fact that these switches intercept TCP traffic that is directed at port 80 and send them all other traffic directly to the WAN router.

## 2.2 Cooperative Caching-Swalla architecture

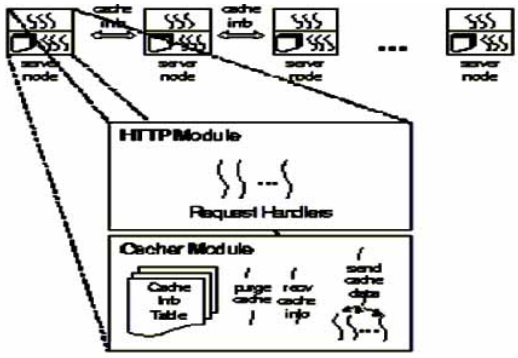


Figure 4: The Swalla architecture.

A different approach to improving Web access performance is presented in [6], by recognizing that processor utilization rather than network bandwidth is the bottleneck in Web sites accessing. This applies especially to sites making extensive use of requests for dynamic content , such as CGI requests. The solution is a distributed Web server, called Swalla, which cooperatively caches the results of CGI requests. Swalla is a multi-threaded, distributed Web Server that runs on a cluster of workstations and shares cache information and cache data between nodes. The server saves the execution results of CGI programs and stores information (meta-data) about the cached data in the cache directory. Each node communicates with each others to exchange cache data and meta-data. As the components of the Swalla architecture illustrated in Figure 4, every Swalla node contains two primary runtime modules, the HTTP module and the cache module.

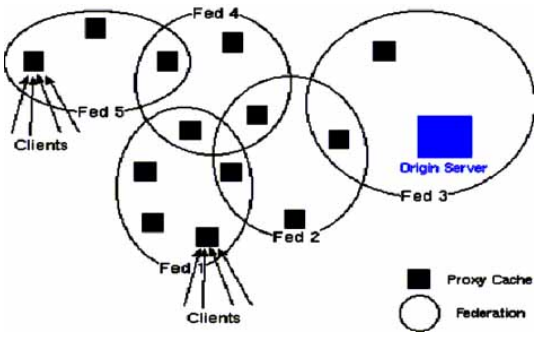


Figure 5: Adaptive Caching configuration.

### 2.3 Adaptive web caching

Authors of [8] argued that an adaptive, highly scalable, and robust web caching system is needed to effectively handle the exponential growth and extreme dynamic environment of the World Wide Web. The system must evolve towards a more scalable, adaptive, efficient, and self-configuring web-caching system in order to effectively support the phenomenal growth in demand for web content on the Internet. The adaptive web caching system provides an effective evolutionary step towards the above goal. Adaptive caching consists of multiple, distributed caches which dynamically join and leave cache groups based on content demand. The general architecture of the envisioned adaptive web caching system would be comprised of many cache servers that self-organize themselves into a tight mesh of overlapping multicast groups and adapt themselves as necessary to changing conditions. This mesh of overlapping groups forms a scalable, implicit hierarchy that is used to efficiently diffuse popular web content towards the demand. There are two main components, which are the underlying communication paths between neighbouring caches and the flow of requests for data along paths. This scheme of architecture is illustrated in Figure 5.

Adaptive caching uses the Cache Group Management Protocol (CGMP) and the Content Routing Protocol (CRP). CGMP specifies how meshes are formed and how individual caches join and leave those meshes. CRP is used to locate cached content from within the existing meshes.

### 2.4 Push Cashing

The idea of having a server decide when and where to cache its documents, was introduced as push-cashing in [9]. The key idea behind this architecture is to keep cached data close to those clients requesting that information. Data is dynamically mirrored as the originating server identifies where requests originate. For example, if a traffic to a west coast based site started to rise because of increasing requests from the coast, the west coast site would respond by initiating an east coast based cache. One main assumption of push cashing is the ability to launch caches that may cross administrative boundaries. Finally, push caching is targeted mostly at content providers, which will most likely control the potential sites at which the caches will be deployed.

### 2.5 Active caching

An active cache scheme is proposed in [10] to support caching of dynamic contents at Web proxies. The growth of the Internet and the World Wide Web has significantly increased the amount of online information and services available to the general population of the society. The Active Cache is a scheme which migrates parts of server processing on each user request to the caching proxy in a flexible, on demand fashion via "cache applets". A cache applet is a server-supplied code that is attached with a URL or a collection of URLs. The code is typically written in a platform independent programming such as Java. Adaptive cache uses applets, located in the cache, to customize objects that could otherwise not cached.

## 3 Proxy Cache Performance

A proxy cache is a link between clients' browsers and Web servers on the Internet. When a user requests a Web document, the request goes through a proxy. If the document is in the proxy cache (cache hit) the proxy can immediately respond to the client's respond. If the requested document is not found (a cache miss) the proxy then attempts to retrieve the document from another location such as a peer or parent proxy cache or the origin server. Once the copy of the document has been retrieved the proxy can complete its response to the client. If the document is cacheable (based on information provided by the origin server or determined from the URL) the proxy may decide to add a copy of the document to its cache. However, if at some point the space required to store all the documents being cached exceeds the available space, the proxy will need to replace a document from the cache. In general, cache replacement algorithms attempt to maximize the hit ratio (the percentage of requests successfully fulfilled by the cache) by holding onto the items most likely to be requested in the future. Unfortunately, recent results suggest that the maximum cache hit rate that can be achieved by any caching algorithm is usually no more than 40% to 50%. This means that one out of two documents can not be found in the cache. The problem is that there is not program which can predict the future user needs. Several solutions have been proposed in the past such as the Top-10 approach to prefetching the Web [11] and the intelligent web caching by using document life histories [3].

Finally, it is useful to evaluate the performance of proxy caching, both for consumers selecting the appropriate system for a particular situation and also for developers working on alternative caching mechanisms.

### 3.1 Performance Metrics and Factors

In this section we will refer to the main performance metrics and the main factors which affect the Web cache performance.

#### 3.1.1 Performance Metrics

Several metrics are commonly used when evaluating web cache performance. The hit rate (HR), as we referred above, is generally the ratio of documents obtained through using the cashing mechanism versus

the total documents requested. A high HR reflects an effective cache policy. If the documents are homogeneous in size, this measure may be a reasonable measure of effectiveness. If the results are of varying sizes, byte hit rate is a better performance measurement. Byte hit rate is defined as the ratio of the number of bytes loaded from the Cache to the total number of bytes accessed. Bandwidth utilization is another measure where the obvious objective is to reduce the amount of bandwidth consumed. A fourth measure is user response time (i.e. the time a user waits for the system to retrieve a requested document). Other measures of cache efficiency include cache server CPU and I/O system utilization, the fraction of total available CPU cycles or disk and object retrieval latency (or page load time), which is especially of interest to end users. Latency is inversely proportional to object hit rate because a cache hit can be served more quickly than a request that must pass through the cache to an origin server and back. Increasing the hit rate does not mean that minimize latency. Caching a few documents with high download latency might actually reduce average latency more then caching many low latency documents. End user latency is difficult to measure at the cache and can be significantly affected by factors outside the cache.[3, 4, 13]

Although these measures are related, optimizing one measurement may not optimize another. For example an increase in HR does not mean that it will necessarily reduce the network traffic.

#### 3.1.2 Performance factors

Various factors affect Web cache performance. The behavior of the user population that request documents from the cache is characterized by the user-access pattern. If a user accesses a small number of documents most of the time then these documents are obvious candidates for caching. Use-access patterns are usually not static and this implies that an effective cache policy should not be static. There are cache replacement policies which decide which document to remove when the cache is full. The cache removal period dictates at what point in time may a document (or documents ) be removed. A continuous removal period implies that documents will be removed when there is no space in the cache to hold the active document. The active document is the document currently being accessed. A fixed cache removal period indicates that documents will only be removed at the beginning of the removal period. Cache size is another factor influencing cache performance. The larger the cache size is the more documents it can maintain and the higher the cache hit ratio is. But, cache space is expensive. Therefore, an optimal cache size involves a trade off between cache cost and cache performance. Document size is also associated with cache performance. Given a certain cache size, the cache can store more small sized documents or fewer large sized documents. Maximum cacheable document size is a user-defined factor that places a ceiling on the size of documents that are allowed to be stored in the cache. Furthermore, there are two others factors which are coorperation and consistency. Coorperation refers to the coordination of users requests among many proxy caches in a hierarchical proxy cache environment. Cache consistency refers to maintaining copies of documents in cache that are not outdated. There are also factors which indirectly affect proxy cache performance such as protection copyright, which increases the complexity of proxy cache design. Finally uncacheable documents are a potential concern.[12]

### 3.2 Evaluating Proxy Cache Performance

It is important to evaluate the performance of proxy cache because it is the best way to recognize drawbacks with particular implementations. Several approaches have been suggested for evaluating proxy cache performance.[5, 12, 14, 15, 16]

In this section, we firstly present the main characteristics which impact proxy performance and cache replacement policies according to [17] and secondly we present the appropriate evaluation mechanisms for proxy systems.

#### 3.2.1 Workload Characterization

In order for Web caching to improve performance it is vital that most objects must be cacheable. Another characteristic is the object set size. Due to the extremely large object set size the proxy cache must be able to quickly determine whether a requested object is cached to reduce response latency. The proxy must be also efficiently update its state on a cache hit, miss or replacement. Furthermore object sizes is another characteristic which impact proxy performance and cache replacement policies. One of the main obstacles for Web caching is working effectively with variable-sized objects. The issue for the proxy cache is to decide whether to cache a large number of small objects (which could increase the hit rate) or to cache a few large objects (possibly increasing the byte hit rate). The recent of reference is another characteristic of Web proxy workloads. It means that objects which have recently been referenced are likely to be re-referenced in the near future. Studies in [11, 17] have found that one-third of all re-references to an object occurred within one hour of the previous reference to the same subject. Approximately two-thirds of re-references occurred within a day of the previous requests. Of course in the Web, it is well known, that "popular documents are very popular". Several recent studies (e.g. [18]) have found that some Web objects are more popular than others. This suggests that popularity, or frequency of reference is a characteristic that could be considered in a cache replacement decision. One final characteristic that could impact Proxy cache replacement decisions is turnover in the active set of objects (the set of objects that users are currently interesting in). Over time the active set changes and objects that were once popular are no longer requested. So, these inactive objects must be removed from the cache to make space for available for new objects that are now in the active set.

#### 3.2.2 Evaluation Methods for Proxy Systems

The most commonly used cache evaluation method according to [5] is that of simulation on a benchmark log of object requests. The byte and page hit rate savings can then be calculated as estimates of latency improvements. In a few cases an artificial dataset with the necessary characteristics, such as appropriate average and median object sizes or similar long-tail distributions of object sizes and object repetitions, is used. More commonly, actual client request logs

are used since they arguably better represent likely request patterns and include exact information about object sizes and retrieval times. Simulation is the simplest mechanism for evaluation as it does not require full implementation. But, simulating the caching algorithm required detailed knowledge of the algorithms which is not always possible, especially for commercial implementations.

# 4 CONCLUSION

Web caching is the best solution to reduce the internet traffic and bandwidth consumption. It is also a low cost technique for improving the Web latency. Nowdays, proxy caches are increasingly used around the world to reduce bandwidth and make less severe delays associated with delays. Web proxy servers sharing their cache directories through a common mapping service that can be queried with at most one message exchange. In this survey we have described the most common architectures which deal with WWW cashing, giving more emphasis on proxy caching scheme. By considering, that it is useful to be able to assess the performance of proxy caches, we have presented the basic metrics and factors for evaluating proxy cache performance. In this survey we have also discussed about the workload characterization providing the most common evaluation methods for proxy systems.

# References


[1] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A, Caching proxies: limitations and potentials,In Proceedings of the 4th International WWW Conference, Boston, MA, December 1995.

[2] Junho Shim, Peter Scheuermann, and Radek Vingralek, Proxy cache design: Algorithms, implementation and performance,IEEE Transactions on Knowledge and Data Engineering, 1999.

[3] Mike Reddy, Graham P.Fletcher, Intelligent web caching using document life histories: A comparison with existing cache management techniques,University of Glamorgan.

[4] John Dilley, Martin Arlitt, Improving Proxy Cache Performance:Analysis of Three Replacement, November 1999.

[5] Brian D. Davison, A survey of proxy cache evaluation techniques,In Proceedings of the 4th International Web Caching Workshop, April 1999.

[6] Vergard Holmedahl, Ben Smith, Tao Yang, Cooperative Caching of Dynamic Content on a Distributed Web Server,University of California.

[7] Barish, Greg, Obraczka, Katia, World Wide Web caching: trends and techniques,IEEE Communications Magazine, Volume 38, Issue 5,2000, pp. 178-185.

[8] S. Michael, K.Nguyen, A.Rosenstein, L. Zhang, S. Floyd and V. Jacobson, Adaptive Web Caching: towards a new global caching architecture,In Proceedings of the Symposium on Internet Technologies and Systems, 1997.

[9] J. Gwertzaman and M. Seltzer, The case for geographical push cashing, Fifth Annual Workshop on Hot Operating Systems, 1995.

[10] P. Cao, J. Zhang, and Kevin Beach, Active cache: Caching dynamic contents on the web,In Proceedings of the 3th International Conference on Web Caching, 1998.

[11] Evangelos P. Markatos, Catherine E. Chronaki, A Top-10 Approach to Prefetching the Web,In Proceedings of the INET '98 conference, July 1998.

[12] Edward F. Watson, Ying Shi and Ye-Sho Chen, A user-access model-driven approach to proxy cache performance analysis, Decision Support Systems, Volume 25, Issue 4,May 1999, pp. 309-338.

[13] Li Fan, Quinn Jacobson, Pei Cao, and Wei Lin, Web prefetching between low-bandwidth clients and proxies: Potential and performance,In Proceedings of the SIGMETRICS '99 Conference, May 1999.

[14] Martin Arlitt , Rich Friedrich , Tai Jin, Performance Evaluation of Web Proxy Cache Replacement Policies, Notes in Computer Science, Volume 1469.

[15] Carlos Maltzahn and Kathy J. Richardson: *Comparing the performance of CERN's httpd and Squid*,In Proceedings of the 1997 NLANR Web Cache Workshop, June 1997.

[16] Carlos Maltzahn, Kathy Richardson, and Dirk Grunwald, Performance issues of enterprise level Web proxies,In ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, June 1997.

[17] J. Almeida and P.Cao, Measuring proxy performance with the wisconsin proxy benchmark, Technical Report 1373, Computer Sciences Dept, Univ. of Wisconsin-Madison, 1998.

[18] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, World Wide Web caching-the application level view of the internet, IEEE Communications Magazine vol.35 June 1997.

[19] Pei Cao and Sandy Irani, Cost-aware WWW proxy caching algorithms, In Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97), Monterey, CA, December 1997.

[20] Jean-Marc Menaud, Valrie Issarny, and Michel Bantre, Improving effectiveness of Web caching, In Springer Verlag, editor, Recent Advances in Distributed Systems, volume LNCS 1752, 2000.